Theoretical Computer Science 40 (1985) 323–328
North-Holland

NOTE

ON RECURSIVE PATH ORDERING

M.S. KRISHNAMOORTHY
Department of Computer Science, Rensselaer Polytechnic Institute, Troy, NY 12181, U.S.A.

P. NARENDRAN*
Computer Science Branch, G.E. Research and Development Center, Schenectady, NY 12345, U.S.A.

Communicated by R.V. Book
Received October 1984
Revised March 1985

Abstract. The 'Recursive Path Ordering' (RPO) scheme of Dershowitz is a powerful way of extending a partial order on a set of function symbols to a well-founded partial order on their set of terms. We prove that, given a pair of terms, the problem of deciding whether they can be made RPO-comparable, by choosing a partial order on their function symbols, is NP-complete.

Key words. Term rewriting systems, termination, partial order, well-founded, MONOTONE 3SAT.

1. Introduction

The uniform termination property (sometimes called Noetherian property) is crucial to applying the Knuth-Bendix completion procedure [4, 9] to term rewriting systems. Several strategems have been used to show that a term rewriting system is uniformly terminating [1, 3, 6, 7, 9, 10, 12]. (The problem in general has been shown to be undecidable [5].) In this note we examine one of them, namely the recursive path ordering (RPO) scheme [1], which is an elegant way of using a partial order on the set of function symbols to obtain a well-founded partial order on the set of terms. Rewrite rule laboratories such as REVE [11] and RRL [8] use this scheme and various extensions of it. Here, we focus our attention on a problem frequently encountered while using RPO: how do we choose the partial order on function symbols to start with? (The notion of 'incrementality', introduced in [6], is related to this question.) The rewrite rule laboratories mentioned above interact with the user to resolve this. We view this problem in an algorithmic setting and show that, given two terms, choosing a partial order on the set of function symbols such that they become comparable by RPO is NP-complete.

* Partially supported by National Science Foundation Grant MCS-8211621.

2. Definitions and basic results

The reader is referred to [2] for a definition of NP-completeness and related results.

A well-founded partial order $>$ on a set E can be extended to the set of *multisets* on E as follows:

$$M1 \gg M2 \text{ iff } \forall x \in M2 - M1\ \exists y \in M1 - M2 \text{ such that } y > x.$$

Let F be a finite set of function symbols of fixed arity and X be a denumerable set of variables. By $T(F, X)$ we denote the set of all possible terms that can be constructed using F and X. For a term t, $\mathrm{Var}(t)$ denotes the set of all variables that occur in t. For example, $\mathrm{Var}(f(x, y, g(y))) = \{x, y\}$. The *size* of a term s is the number of occurrences of function and variable symbols in s and is denoted by $|s|$.

A *term rewriting system* P over a set of terms $T(F, X)$ is a finite set of rewrite rules of the form $l_i \to r_i$, where $l_i, r_i \in T(F, X)$. We write $t \Rightarrow t'$ to indicate that the term t' can be derived from the term t by a single application of a rule in P to one of its subterms. P is said to be *uniformly terminating* if there exist no infinite sequences of terms $t_i \in T(F, X)$ such that $t_1 \Rightarrow t_2 \Rightarrow \cdots$.

Recursive Path Ordering (RPO) [1] provides us a way to 'lift' a well-founded partial order $>$ on F to a well-founded partial order $>_{\mathrm{rpo}}$ on $T(F, X)$.

Definition 2.1. (a) $s = f(s_1, \ldots, s_m) >_{\mathrm{rpo}} t = x$ if and only if $x \in \mathrm{Var}(s)$.

(b) $s = f(s_1, \ldots, s_m) >_{\mathrm{rpo}} t = g(t_1, \ldots, t_n)$ if and only if

(1) $f > g$ and $s >_{\mathrm{rpo}} t_i$ for all i, $1 \leq i \leq n$, or,

(2) $f = g$ and $\{s_1, \ldots, s_m\} \gg_{\mathrm{rpo}} \{t_1, \ldots, t_n\}$, or,

(3) $f \not\geq g$ and $s_i \geq_{\mathrm{rpo}} t$ for some i, $1 \leq i \leq m$.

Two terms are considered equal if they are *permutationally equivalent* or, in other words, they are the same except for permutations among subterms. For example, $f(x, y)$ is equal to $f(y, x)$ by this definition.

Theorem 2.2 ([1]). *Let $P = \{l_i \to r_i \mid 1 \leq i \leq n\}$ be a term rewriting system on $T(F, X)$ and $>$ a partial order on F. If $l_i >_{\mathrm{rpo}} r_i$, for $i = 1, \ldots, n$ then P is uniformly terminating.*

The following observations are obvious:

(i) if t is a proper subterm of s, then $s >_{\mathrm{rpo}} t$,

(ii) if $\mathrm{Var}(s)$ and $\mathrm{Var}(t)$ are incomparable, then s and t are incomparable by RPO,

(iii) if $\mathrm{Var}(s)$ is a proper subset of $\mathrm{Var}(t)$, then it cannot be that $s >_{\mathrm{rpo}} t$.

Now, when we use the Knuth-Bendix procedure, we do not usually have a set of rewrite rules to start with; we only have a bunch of equations, each of which we have to orient into a rewrite rule in such a way that the resulting term rewriting system is uniformly terminating. Given a well-founded order $>$ on the terms, an equation $s = t$ can be made into the rewrite rule $s \to t$ only when $s > t$ and into $t \to s$

only when $t > s$. This motivates us to define the *RPO comparison problem* (RPOCP) as follows:

INSTANCE: A pair of terms s and t.

QUESTION: Are s and t *RPO-comparable*, i.e., is there a partial order $>$ on the set of function symbols F such that either $s >_{\text{rpo}} t$ or $t >_{\text{rpo}} s$?

Note that this problem is trivial if s and t are ground terms, or if s and t are monadic terms, since any total order on function symbols will make them comparable.

Lemma 2.3. *Given a partial order $>$ on the set of function symbols F and two terms s and t, we can determine whether $s >_{\text{rpo}} t$ or $s =_{\text{rpo}} t$ making $O(|s| \cdot |t|)$ comparisons using the given partial order $>$.*

Proof. In fact, we can show that the number of comparisons is bounded above by $|s| \cdot |t|$ itself. We proceed to show this by induction.

Basis step: $|s| = |t| = 1$. Clearly, we only need one comparison.

Induction step: Let $s = f(s_1, \ldots, s_m)$ and $t = g(t_1, \ldots, t_n)$.

(i) $f > g$. Clearly, $s \neq t$. Now, to determine whether $s >_{\text{rpo}} t$, s must be compared with each t_i, $1 \leq i \leq n$. By induction hypothesis, each of them will need at most $|s| \cdot |t_i|$ comparisons. Thus, an upper bound on the total number of steps is

$$1 + |s| \cdot \left(\sum_{i=1}^{n} (|t_i|) \right) \leq |s| \cdot |t|.$$

(ii) $f = g$. Here, we have the additional possibility of s being equal to t (under RPO). But, in any case, it is enough to compare the multisets $\{s_1, \ldots, s_m\}$ and $\{t_1, \ldots, t_n\}$. While comparing these multisets, each s_i, $1 \leq i \leq m$, will have to be compared with each t_j, $1 \leq j \leq n$, in the worst case. This will take at most $\sum_{i=1}^{m} (\sum_{j=1}^{n} (|s_i| \cdot |t_j|))$ comparisons. Again,

$$\sum_{i=1}^{m} \left(\sum_{j=1}^{n} (|s_i| \cdot |t_j|) \right) < \sum_{i=1}^{m} (|s_i| + 1) \cdot \sum_{j=1}^{n} (|t_j| + 1)$$

and we are done.

(iii) $f \not> g$. In this case the proof is similar to that of case (i) above. □

The above lemma enables us to state the following corollary.

Corollary 2.4. *Given a partial order $>$ on the set of function symbols F and two terms s and t, we can determine whether $s >_{\text{rpo}} t$ in time polynomial in $|s|$ and $|t|$.*

Proof. By a proper choice of data structure we can design an algorithm that runs in time $O(|s| \cdot |t|)$. The details are left to the reader. □

Theorem 2.5. RPOCP *is in* NP.

Proof. Given two terms s and t, one can nondeterministically choose a partial order $>$ and, using $>$, compare s and t in time polynomial in $|s| \cdot |t|$. □

3. Proof of NP-hardness

An instance $C_1 \wedge C_2 \wedge \cdots \wedge C_m$ of 3SAT is said to be in MONOTONE 3SAT form if

(i) all the literals in each clause are either positive or negative, and
(ii) the number of literals in any clause equals three.

Theorem 3.1 ([2]). *The satisfiability of formulae in* MONOTONE 3SAT *form is* NP-*complete.*

We now proceed to reduce MONOTONE 3SAT to the RPO-comparison problem. Let $\mathrm{PF} = C_1 \wedge C_2 \wedge \cdots \wedge C_m$ be a formula in MONOTONE form. For each propositional variable x_i in PF, we have a binary function symbol f_i. We also have an $(m+1)$-ary function ϕ and another binary function h. Our intention is to simulate $x_i = 1$ by $f_i > h$ and $x_i = 0$ by $f_i < h$.

Lemma 3.2. *Let* $s_1 = f_1(f_2(f_3(x, x), x), x)$ *and* $t_1 = h(h(h(h(x, x), x), x), x)$. *Now,*

$$s_1 >_{\mathrm{rpo}} t_1 \text{ iff } (f_1 > h) \text{ or } (f_2 > h) \text{ or } (f_3 > h).$$

Proof. The proof is obvious from the definition of RPO. □

Lemma 3.3. *Let* $s_2 = h(f_1(h(f_3(x, x), x), f_2(h(x, x), x)), f_2(f_3(x, x), x)$ *and* $t_2 = f_1(f_2(f_3(x, x), x), x)$. *Then,*

$$s_2 >_{\mathrm{rpo}} t_2 \text{ iff } (h > f_1) \text{ or } (h > f_2) \text{ or } (h > f_3).$$

Proof. The 'if' part follows from the definition of RPO. Now if $h \not> f_1$, then, either, $f_1(h(f_3(x, x), x), f_2(h(x, x), x)) \geq t_2$, or, $f_2(f_3(x, x), x) \geq t_2$. The latter is clearly impossible, since $f_2(f_3(x, x), x)$ is a proper subterm of t_2. Furthermore, $f_1(h(f_3(x, x), x), f_2(h(x, x), x)) \neq t_2$. Therefore, it must be that $f_1(h(f_3(x, x), x), f_2(h(x, x), x) > t_2$ and this necessitates, either, $h(f_3(x, x), x) > f_2(f_3(x, x), x)$, or $f_2(h(x, x), x) > f_2(f_3(x, x), x)$. It can now be easily seen that we need $h > f_2$ in the former case and $h > f_3$ in the latter. □

From the above two lemmas it is clear how each clause C in PF can be simulated by a pair of terms p and q such that $p >_{\text{rpo}} q$ if and only if C is satisfiable. Let $((s_1, t_1), (s_2, t_2), \ldots, (s_m, t_m))$ be such that

(i) (s_i, t_i) simulates clause C_i,

(ii) for all i, j such that $i \neq j$, (s_i, t_i) and (s_j, t_j) are standardized apart (i.e., have distinct sets of variables).

Further, let $s = \phi(s_1, \ldots, s_m, z)$ and $t = \phi(t_1, \ldots, t_m, y)$ where y is a variable occurring in one of the s_i's and z is a new variable. Since z does *not* occur in t, it cannot be that $t >_{\text{rpo}} s$. Hence, the partial order $>$ on the function symbols should be such that $s >_{\text{rpo}} t$. But $s >_{\text{rpo}} t$ if and only if

$$\{s_1, \ldots, s_m, z\} \gg \{t_1, \ldots, t_m, y\}. \tag{1}$$

Note that every t_i, $1 \leq i \leq m$, can be compared only with the corresponding s_i since terms with distinct sets of variables cannot be compared. Of course, $y < s_i$ for some i, where y occurs in s_i. Hence, the relation (1) can be satisfied if and only if

$$s_i > t_i \quad \text{for } 1 \leq i \leq m.$$

Thus, since the transformation from an instance of MONOTONE 3SAT to an instance of RPOCP is in polynomial time, we have the following theorem.

Theorem 3.4. *Given the equation $s = t$, checking whether there exists a partial order $>$ on the function symbols that occur in s and t such that the equation can be oriented by* RPO *is* NP*-complete.*

In conclusion, we would like to mention that the above construction also enables us to prove similar results for extensions of RPO, such as the Recursive Decomposition Ordering of [6] and the Path Ordering of [7].

Acknowledgment

We thank Chuck Fiduccia and Deepak Kapur for several helpful comments and suggestions.

References

[1] N. Dershowitz, Orderings for term-rewriting systems, *Theoret. Comput. Sci.* **17** (1982) 279–301.

[2] M.R. Garey and D.S. Johnson, *Computers and Intractability* (Freeman, San Francisco, CA, 1979).

[3] J.V. Guttag, D. Kapur and D.R. Musser, On proving uniform termination and restricted termination of rewriting systems, *SIAM J. Comput.* **12** (1) (1983) 189–214.

[4] G. Huet, Confluent reductions: Abstract properties and applications to term rewriting systems, *J. ACM* **27** (4) (1980) 797–821.

[5] G. Huet and D.S. Lankford, On the uniform halting problem for term rewriting systems, Rapport Laboria 283, INRIA, Paris, 1978.

[6] J.-P. Jouannaud, P. Lescanne and F. Reinig, Recursive decomposition ordering, in: D. Bjørner, ed., *Formal Description of Programming Concepts - II* (North-Holland, Amsterdam, 1983) 331-348.

[7] D. Kapur, P. Narendran and G. Sivakumar, A path ordering for proving termination of term rewriting systems, *Proc. Colloquium on Trees in Algebra and Programming* (CAAP'85), Berlin, 1985.

[8] D. Kapur and G. Sivakumar, Architecture of and experiments with RRL, a rewrite rule laboratory, *Proc. NSF Workshop on Rewrite Rule Laboratory*, Rensselaerville, NY, 1983.

[9] D.E. Knuth and P.B. Bendix, Simple word problems in universal algebras, in: J. Leech, ed., *Computational Problems in Abstract Algebras* (Pergamon Press, Oxford, 1970) 263-297.

[10] D.S. Lankford, On proving term rewriting systems are noetherian, Memo MTP-3, Mathematics Dept., Louisiana Technical Univ., Ruston, LA, 1979.

[11] P. Lescanne, Computer experiments with the REVE term rewriting system generator, *10th POPL Conf.*, Austin, TA, 1983.

[12] D.A. Plaisted, A recursively defined ordering for proving termination of term rewriting systems, Rept. R78-943, Dept. of Computer Science, Univ. of Illinois, Urbana, IL.